Percentage With Cap Argument

1 Preliminaries

Basic notation. For two integers $n < m$, we write $[n, m]$ to denote the set $\{n, n+1, \ldots, m\}$. When $n = 1$, we simply write $[m]$ to denote the set $\{1, \ldots, m\}$. For any finite set S, we use $x \leftarrow_{\mathsf{R}} S$ to denote the process of sampling an element $x \in S$ uniformly at random. Unless specified otherwise, we use λ to denote the security parameter. We say that an algorithm is efficient if it runs in probabilistic polynomial time in the length of its input. We say that a function $f : \mathbb{N} \to \mathbb{N}$ is negligible if $f = o(1/n^c)$ for any positive integer $c \in \mathbb{N}$. Throughout the exposition, we use $\mathsf{poly}(\cdot)$ and $\mathsf{negl}(\cdot)$ to denote any polynomial and negligible functions respectively.

1.1 Discrete Log Relation Assumption

The discrete log relation assumption states that given a number of random group elements in $\mathbb{G}$, no efficient adversary can find a non-trivial relation on these elements.

Definition 1.1 (Discrete Log Relation)**.** Let $\mathbb{G} = \mathbb{G}(\lambda)$ be a group of prime order p. Then the *discrete log relation* assumption on $\mathbb{G}$ states that for any efficient adversary $\mathcal{A}$ and $n \geq 2$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\Pr\left[\mathcal{A}(G_1, \ldots, G_n) \to a_1, \ldots, a_n \in \mathbb{Z}_p : \exists\ a_i \neq 0 \wedge \sum_{i \in [n]} a_i \cdot G = 0\right] = \mathsf{negl}(\lambda),$$

where $G_1, \ldots, G_n \leftarrow_{\mathsf{R}} \mathbb{G}$.

1.2 Rewinding Lemma

To prove security, we make use of the rewinding lemma. For the purpose of this document, we do not require the rewinding lemma in its full generality and therefore, we rely on the following simple variant from the work of Boneh et al. [1].

Lemma 1.2 (Rewinding Lemma)**.** *Let S, R, and T be finite, non-empty sets, and let X, Y, Y', Z, and Z' be mutually independent random variables such that*

- *X takes values in the set S,*
- *Y and Y' are each uniformly distributed over R,*
- *Z and Z' take values in the set T.*

Then for any function $f : S \times R \times T \to \{0, 1\}$, we have

$$\Pr\left[f(X, Y, Z) = 1 \wedge f(X, Y', Z') = 1 \wedge Y \neq Y'\right] \geq \varepsilon^2 - \varepsilon/N,$$

where $\varepsilon = \Pr[f(X, Y, Z) = 1]$ and $N = |R|$.

2 Zero-Knowledge Argument Definitions

In full generality, zero-knowledge argument systems can be defined with respect to any class of decidable languages. However, to simplify the presentation, we define argument systems with respect to CRS-dependent languages. Specifically, let $\mathcal{R} \subset \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^*$ be an efficiently decidable ternary relation. Then a CRS-dependent language for a string $\rho \in \{0,1\}^*$ is defined as

$$\mathcal{L}_\rho = \{u \mid \exists\ w : (\rho, u, w) \in \mathcal{R}\}.$$

We generally refer to ρ as the common reference string, u as the instance of the langauge, and w as the witness for u.

For a class of CRS-dependent languages, an argument system consists of the following algorithms.

Definition 2.1 (Argument System)**.** A non-interactive argument system Π_{AS} for a CRS-dependent relation $\mathcal{R}$ consists of a tuple of efficient algorithms $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda) \to \rho$: On input the security parameter λ, the setup algorithm returns a common reference string ρ.
- $\mathcal{P}(\sigma, u, w)$: The prover $\mathcal{P}$ is an interactive algorithm that takes in as input a common reference string σ, instance u, and witness w. It interacts with the verifier $\mathcal{V}$ according to the specification of the protocol.
- $\mathcal{V}(\sigma, u)$: The verifier $\mathcal{V}$ is an interactive algorithm that takes in as input a common reference string ρ and an instance x. It interacts with the prover $\mathcal{P}$ in the protocol and in the end, it either accepts (returns 1) or rejects (returns 0) the instance x.

We use $\langle \mathcal{P}(\rho, u, w), \mathcal{V}(\rho, u) \rangle = 1$ to denote the event that the verifier $\mathcal{V}$ accepts the instance of the protocol. We use $\langle \mathcal{P}(\rho, u, w), \mathcal{V}(\rho, u) \rangle \to \mathsf{tr}$ to denote the communication transacript between the prover $\mathcal{P}$ and verifier $\mathcal{V}$ during a specific execution of the protocol.

An argument system must satisfy a correctness and two security properties. The correctness property of an argument system is generally referred to as *completeness*. It states that if the prover $\mathcal{P}$ takes in as input a valid instance-witness tuple $(\rho, u, w) \in \mathcal{R}$ and follows the protocol specification, then it must be able to convince the verifier to accept.

Definition 2.2 (Completeness)**.** Let Π_{AS} be a proof system for a relation $\mathcal{R}$. Then we say that Π_{AS} satisfies perfect completeness if for any $(u, w) \in \mathcal{R}$, we have

$$\Pr\left[\langle \mathcal{P}(\rho, u, w), \mathcal{V}(\rho, u) \rangle = 1\right] = 1,$$

where $\rho \leftarrow \mathsf{Setup}(1^\lambda)$.

The first security property that an argument system must satisfy is *soundness*, which can be defined in a number of ways. In this work, we work with *computational witness-extended emulation* as presented in Bulletproofs [2].

Definition 2.3 (Soundness [3, 4, 2])**.** Let Π_{AS} be a proof system for a relation $\mathcal{R}$. Then we say that Π_{AS} satisfies *witness-extended emulation* soundness if for all deterministic polynomial time $\mathcal{P}^*$,

there exists an efficient emulator $\mathcal{E}$ such that for all efficient adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\left| \begin{array}{c} \Pr\left[\mathcal{A}_2(\mathsf{tr}) = 1 \;\middle|\; \begin{array}{c} \rho \leftarrow \mathsf{Setup}(1^\lambda), (u, \mathsf{st}) \leftarrow \mathcal{A}_1(\rho), \\ \mathsf{tr} \leftarrow \langle \mathcal{P}^*(\rho, u, \mathsf{st}), \mathcal{V}(\rho, u) \rangle \end{array}\right] - \\ \Pr\left[\mathcal{A}_2(\mathsf{tr}) = 1 \wedge (\mathsf{tr} \text{ accepting } \Rightarrow (\rho, u, w) \in \mathcal{R}) \;\middle|\; \begin{array}{c} \rho \leftarrow \mathsf{Setup}(1^\lambda), \\ (u, \mathsf{st}) \leftarrow \mathcal{A}_1(\rho), \\ (\mathsf{tr}, w) \leftarrow \mathcal{E}^{\mathcal{O}}(\rho, u) \end{array}\right] \end{array} \right| = \mathsf{negl}(\lambda),$$

where the oracle is defined as $\mathcal{O} = \langle \mathcal{P}^*(\rho, u, \mathsf{st}), \mathcal{V}(\rho, u) \rangle$. The oracle $\mathcal{O}$ allows the emulator $\mathcal{E}$ to rewind the protocol to a specific point and resume the protocol after reprogramming the verifier with fresh randomness.

Traditionally, the soundness condition for an argument system of knowledge requires that there exists an extractor that can use its rewinding capability to extract a valid witness from any accepting transcript of the protocol that is produced by a dishonest prover $\mathcal{P}^*$. The witness-extended emulation strengthens this traditional definition by requiring that the extractor (emulator) not only successfully extracts a valid witness, but also produces (emulates) a valid transcript of the protocol for which the verifier accepts. The value st in the definition above can be viewed as the internal state of $\mathcal{P}^*$, which can also be its randomness.

The second security property that we require from an argument system is the zero-knowledge property. All argument systems that we rely on in the `ZK-Token` program are public coin protocols that we ultimately convert into a non-interactive protocol. Therefore, we rely on the standard zero-knowledge property against honest verifiers.

Definition 2.4 (Zero-Knowledge)**.** Let Π_{AS} be a proof system for a relation $\mathcal{R}$. Then we say that Π_{AS} satisfies *honest verifier* zero-knowledge if there exists an efficient simulator $\mathcal{S}$ such that for all efficient adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we have

$$\Pr\left[(\rho, u, w) \in \mathcal{R} \wedge \mathcal{A}_1(\mathsf{tr}) = 1 \;\middle|\; \begin{array}{c} \rho \leftarrow \mathsf{Setup}(1^\lambda), (u, w, \tau) \leftarrow \mathcal{A}_2(\rho), \\ \mathsf{tr} \leftarrow \langle \mathcal{P}(\rho, u, w), \mathcal{V}(\rho, u; \tau) \rangle \end{array}\right]$$

$$= \Pr\left[(\rho, u, w) \in \mathcal{R} \wedge \mathcal{A}_1(\mathsf{tr}) = 1 \;\middle|\; \begin{array}{c} \rho \leftarrow \mathsf{Setup}(1^\lambda), \\ (u, w, \tau) \leftarrow \mathcal{A}_2(\rho), \\ \mathsf{tr} \leftarrow \mathcal{S}(u, \tau) \end{array}\right],$$

where ρ is the public coin randomness used by the verifier.

3 Argument System Description

A percentage-with-cap argument is a more specialized zero-knowledge argument system than other argument systems in the ZK ElGamal Proof program. It is most effectively used in combination with range proofs and is motivated by transfer fees in programs like the confidential transfer extension of the Solana token-2022 program.

Background on Fees. When a mint in the Solana token-2022 program is extended for fees, every transfer incurs a transfer fee that is withheld in accounts to be transferred to a designated fee authority. A fee is determined by two parameters: the fee percentage basis point bp and the maximum fee parameter $\mathsf{max}_{\mathsf{fee}}$.

- bp: The basis point parameter is a positive integer that specifies the fee percentage rate in increments of 0.01%. If the fee does not result in an integer, then it is rounded up to the nearest integer.
- $\mathsf{max_{fee}}$: The maximum fee parameter specifies the cap on the fees. If the fee that is calculcated with respect to the basis point results in a value that is greater than $\mathsf{max_{fee}}$, then the fee is set to $\mathsf{max_{fee}}$.

For example, consider the case when $\mathsf{bp} = 200$ and $\mathsf{max_{fee}} = 3$. If the transfer amount is 100, then the fee is 2, which is 2% of the transfer amount. If the transfer amount is 200, then the fee is 3 since 2% of 200 (which is 4) is greater than the cap value of 3.

3.1 Intuition

When a mint is extended for transfer fees, the program requires the sender to include Pedersen commitments for both the transfer amount $\mathsf{Comm_{tx\text{-}amt}}$ and the transfer fee $\mathsf{Comm_{fee}}$ in the instruction data. Additionally, the sender must provide a zero-knowledge argument that $\mathsf{Comm_{fee}}$ holds a valid fee for the amount committed in $\mathsf{Comm_{tx\text{-}amt}}$. For clarity, let us consider two distinct cases.

Case 1: Percentage Fee Less than Max Fee. This case covers transfers where the fee calculated from the basis points is less than $\mathsf{max_{fee}}$. If the fee calculation does not require rounding, the homomorphic property of Pedersen commitments means $\mathsf{Comm_{tx\text{-}amt}}$ and $\mathsf{Comm_{fee}}$ must satisfy the relation $\mathsf{Comm_{tx\text{-}amt}} \cdot (\mathsf{bp}/10000) = \mathsf{Comm_{fee}}$ or equivalently

$$\mathsf{Comm_{fee}} \cdot 10000 - \mathsf{Comm_{tx\text{-}amt}} \cdot \mathsf{bp} = 0.$$

If there is rounding involved, then the value $\mathsf{Comm_{fee}} \cdot 10000 - \mathsf{Comm_{tx\text{-}amt}} \cdot \mathsf{bp}$ must be in the range $[0, 10000)$. We refer to the difference $\mathsf{Comm_{fee}} \cdot 10000 - \mathsf{Comm_{tx\text{-}amt}} \cdot \mathsf{bp}$ as the *delta fee*.

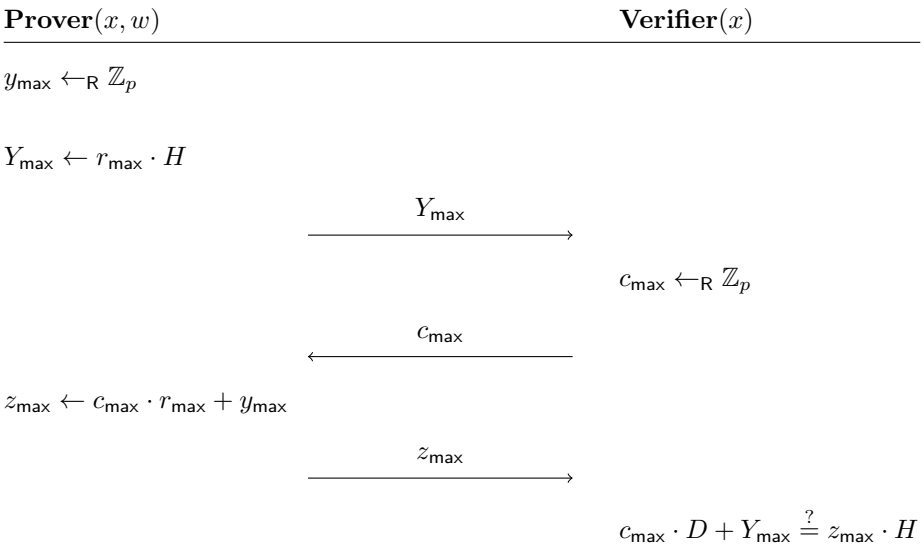
In this case, it is sufficient to require that the sender of a confidential transfer include a range proof to certify that $\mathsf{Comm_{fee}} \cdot 10000 - \mathsf{Comm_{tx\text{-}amt}} \cdot \mathsf{bp}$ is in the range $[0, 10000)$. Although this can be proved directly on the commitment $\mathsf{Comm_{fee}} \cdot 10000 - \mathsf{Comm_{tx\text{-}amt}} \cdot \mathsf{bp}$, in order to utilize the OR proof discussed below, we require that the sender include a fresh commitment to the delta fee $\mathsf{Comm_{delta}}$ along with a commitment-commitment equality argument certifying that $\mathsf{Comm_{fee}} \cdot 10000 - \mathsf{Comm_{tx\text{-}amt}} \cdot \mathsf{bp}$ and $\mathsf{Comm_{delta}}$ are commitments to the same value.

Below, we provide the details of the commitment-commitment equality argument system. For the argument description, we refer to C_{delta} as the group element corresponding to the commitment $\mathsf{Comm_{fee}} \cdot 10000 - \mathsf{Comm_{tx\text{-}amt}} \cdot \mathsf{bp}$, and we refer to C_{claimed} as the group element corresponding to the commitment $\mathsf{Comm_{delta}}$.

Prover(x, w)		**Verifier**(x)
$y_x \leftarrow_{\mathsf{R}} \mathbb{Z}_p$		
$y_{\mathsf{delta}} \leftarrow_{\mathsf{R}} \mathbb{Z}_p$		
$y_{\mathsf{claimed}} \leftarrow_{\mathsf{R}} \mathbb{Z}_p$		
$Y_{\mathsf{delta}} \leftarrow y_x \cdot G + y_{\mathsf{delta}} \cdot H$		
$Y_{\mathsf{claimed}} \leftarrow y_x \cdot G + y_{\mathsf{claimed}} \cdot H$		
	$\xrightarrow{Y_{\mathsf{delta}}, Y_{\mathsf{claimed}}}$	
		$c_{\mathsf{equality}} \leftarrow_{\mathsf{R}} \mathbb{Z}_p$
	$\xleftarrow{c_{\mathsf{equality}}}$	
$z_x \leftarrow c_{\mathsf{equality}} \cdot x + y_s$		
$z_{\mathsf{delta}} \leftarrow c_{\mathsf{equality}} \cdot r_{\mathsf{delta}} + y_{\mathsf{delta}}$		
$z_{\mathsf{claimed}} \leftarrow c_{\mathsf{equality}} \cdot r_{\mathsf{claimed}} + y_{\mathsf{claimed}}$		
	$\xrightarrow{z_x, z_{\mathsf{delta}}, z_{\mathsf{claimed}}}$	
		$z_x \cdot G + z_{\mathsf{delta}} \cdot H \stackrel{?}{=} c_{\mathsf{equality}} \cdot C_{\mathsf{delta}} + Y_{\mathsf{delta}}$
		$z_x \cdot G + z_{\mathsf{claimed}} \cdot H \stackrel{?}{=} c_{\mathsf{equality}} \cdot C_{\mathsf{claimed}} + Y_{\mathsf{claimed}}$

At the start of the protocol, the prover holds the real delta value x and the Pedersen openings r_{delta}, r_{claimed} of the commitments C_{delta} and C_{claimed}. The prover starts by sampling three random blinding factors y_x, y_{delta}, and y_{claimed}, and commits to these values in the form of Y_{delta} and Y_{claimed}. Upon receiving a challenge, the prover blinds the x, r_{delta}, and r_{claimed} values in the form of z_x, z_{delta}, and z_{claimed}. When the prover sends these values to the verifier, the verifier checks their consistency.

Case 2: Fee greater than $\mathsf{max}_{\mathsf{fee}}$. Consider the case when the fee is equal to $\mathsf{max}_{\mathsf{fee}}$. In this case, the fee commitment $\mathsf{Comm}_{\mathsf{fee}}$ must be a commitment to $\mathsf{max}_{\mathsf{fee}}$. Proving that $\mathsf{Comm}_{\mathsf{fee}}$ is a commitment to $\mathsf{max}_{\mathsf{fee}}$ is equivalent to proving that $\mathsf{Comm}_{\mathsf{fee}} - \mathsf{max}_{\mathsf{fee}} \cdot G$ is a commitment to zero where G is the generator of the underlying group. This is equivalent zero-balance argument system. Below, we denote $D = \mathsf{Comm}_{\mathsf{fee}} - \mathsf{max}_{\mathsf{fee}} \cdot G$, which must be a real Pedersen commitment to zero: $D = 0 \cdot G + r_{\mathsf{max}} \cdot H = r_{\mathsf{max}} \cdot H$ where H is a fixed group element associated with the Pedersen commitment scheme and r_{max} is an opening to the commitment.



At the start of the protocol, the prover holds the Pedersen openings r_{max}, the commitment D. The prover starts by sampling three random blinding factor y_{max}, and commits to this value in the form of Y_{max}. Upon receiving a challenge, the prover blinds the r_{max}, value in the form of z_r. When the prover sends this value of the verifier, the verifier checks their consistency.

OR-composition. To prevent revealing whether the transfer fee is above or below the maximum, the two sigma proofs are combined using a standard OR composition technique. The prover proves that one of two statements is true, without revealing which one. This is done by honestly generating the proof for the true case and simulating a valid-looking proof for the false case.

- If the the fee is less than the max fee, the prover honestly generate the commitment-commitment equality proof. It then simulates the zero-balance proof by sampling random values for z_{max} and c_{max}, and then solving for the Y_{max} value that satisfies the verification equation.
- If the fee is equal to the max fee, the prover honestly generates the zero-balance proof. It then simulates the commitment-commitment equality proof by sampling random values for z_s, z_x, z_r, and c, and then solving for the Y_{delta} and Y_{claimed} values that satisfy the verification equation.

Specification. We now provide the full specification of the percentage-with-cap argument. The language $\mathcal{L}^{\mathsf{fee}}_{G,H,\mathsf{bp},\mathsf{max}_{\mathsf{fee}}}$ that is captured by the protocol is specified as follows:

- **Instance**: $u = (C_{\mathsf{amt}}, C_{\mathsf{fee}}, C_{\mathsf{claimed}}) \in \mathbb{G}^3$.
- **Witness**: $w = (x_{\mathsf{amt}}, r_{\mathsf{amt}}, x_{\mathsf{fee}}, r_{\mathsf{fee}}, x_{\mathsf{claimed}}, r_{\mathsf{claimed}}) \in \mathbb{Z}_p^3$ where at least one of the following relations hold
  1. $C_{\mathsf{fee}} \cdot 10000 - \mathsf{bp} \cdot C_{\mathsf{amt}} = C_{\mathsf{claimed}}$
  2. $C_{\mathsf{fee}} - \mathsf{max}_{\mathsf{fee}} \cdot G = r_{\mathsf{fee}} \cdot H$

The language $\mathcal{L}^{\mathsf{fee}}_{G,H,\mathsf{bp},\mathsf{max}_{\mathsf{fee}}}$ is specified by two group elements $G, H \in \mathbb{G}$ and the fee parameters bp, $\mathsf{max}_{\mathsf{fee}}$. The group elements C_{amt}, C_{fee}, and C_{claimed} correspond to the Pedersen commitments

to the transfer amount, transfer fee, and "claimed" delta fee. The scalar elements r_{amt}, r_{fee}, and r_{claimed} correspond to the opening of these commitments. The value x_{amt} corresponds to the transfer amount while the value x_{fee} corresponds to the transfer fee.

Protocol. In the protocol description below, we denote $D \in \mathbb{G}$ to denote the commitment $C_{\mathsf{fee}} \cdot G$.

- **Prover**(u, w): The prover proceeds as follows:
  - **Fee less than max fee**:
    1. The prover simulates the zero-balance proof by sampling a random challenge c_{max} and blinded value z_{max} randomly from $\mathbb{Z}_p$ and then computing $Y_{\mathsf{max}} \leftarrow z_{\mathsf{max}} \cdot H - c_{\mathsf{max}} \cdot D$.
    2. Then it proceeds to generate proof for equality proof by first sampling the blinding factors $y_x, y_{\mathsf{delta}}, y_{\mathsf{claimed}} \leftarrow_{\mathsf{R}} \mathbb{Z}_p$. It computes $Y_{\mathsf{delta}} \leftarrow y_x \cdot G + y_{\mathsf{delta}} \cdot H$ and $Y_{\mathsf{claimed}} \leftarrow y_x \cdot G + y_{\mathsf{claimed}} \cdot H$. It sends Y_{delta}, Y_{claimed}, and Y_{max} to the verifier.
    3. Upon receiving a challenge value $c \in \mathbb{Z}_p$ from the verifier, it computes $c_{\mathsf{equality}} \leftarrow c - c_{\mathsf{max}}$.
    4. Then it computes the blinded values $z_x \leftarrow c_{\mathsf{equality}} \cdot x_{\mathsf{amt}} + y_x$, $z_{\mathsf{delta}} \leftarrow c_{\mathsf{equality}} \cdot r_{\mathsf{delta}} + y_{\mathsf{delta}}$, and $z_{\mathsf{claimed}} \leftarrow c_{\mathsf{equality}} \cdot r_{\mathsf{claimed}} + y_{\mathsf{claimed}}$.
    5. The prover then finally provides c_{max}, z_x, z_{delta}, z_{claimed}, and z_{max} to the verifier.
  - **Fee equal to max fee**:
    1. The prover simulates the equality proof by first sampling a random challenge c_{max} and the blinded values z_x, z_{delta}, and z_{claimed} randomly from $\mathbb{Z}_p$. It then computes $Y_{\mathsf{delta}} \leftarrow c_{\mathsf{equality}} \cdot C_{\mathsf{delta}} - z_x \cdot G - z_{\mathsf{delta}} \cdot H$ and $Y_{\mathsf{claimed}} \leftarrow c_{\mathsf{equality}} \cdot C_{\mathsf{claimed}} - z_x \cdot G - z_{\mathsf{claimed}} \cdot H$.
    2. Then it proceeds to generate proof for the zero-balance proof by first sampling the blinding factor $y_{\mathsf{max}} \leftarrow_{\mathsf{R}} \mathbb{Z}_p$ and computes $Y_{\mathsf{max}} \leftarrow r_{\mathsf{max}} \cdot H$. It sends Y_{delta}, Y_{claimed}, and Y_{max} to the verifier.
    3. Upon receiving a challenge value $c \in \mathbb{Z}_p$ from the verifier, it computes $c_{\mathsf{max}} \leftarrow c - c_{\mathsf{equality}}$.
    4. Then it computes the blinded value $z_{\mathsf{max}} \leftarrow c_{\mathsf{max}} \cdot r_{\mathsf{max}} + y_{\mathsf{max}}$ for the zero-balance proof.
    5. Finally, the prover provides c_{max}, z_x, z_{delta}, z_{claimed}, and z_{max} to the verifier.
- **Verifier**(u): The verifier logic is the same independent of whether the fee is the percentage amount or the max fee.
  1. Upon receiving the values Y_{delta}, Y_{claimed}, and Y_{max} from the prover, it samples a random challenge $c \leftarrow_{\mathsf{R}} \mathbb{Z}_p$ and provides c to the prover.
  2. Upon receiving the values c_{max}, z_x, z_{delta}, z_{claimed}, and z_{max} from the prover, the verifier proceeds as follows:
     - It computes the challenge for the equality argument $c_{\mathsf{equality}} \leftarrow c - c_{\mathsf{max}}$.
     - It verifies the equality argument relation

$$z_x \cdot G + z_{\mathsf{delta}} \cdot H \stackrel{?}{=} c_{\mathsf{equality}} \cdot C_{\mathsf{delta}} + Y_{\mathsf{delta}}$$

$$z_x \cdot G + z_{\mathsf{claimed}} \cdot H \stackrel{?}{=} c_{\mathsf{equality}} \cdot C_{\mathsf{claimed}} + Y_{\mathsf{claimed}}$$

– It verifies the zero balance argument relation

$$c_{\mathsf{max}} \cdot D + Y_{\mathsf{max}} \stackrel{?}{=} z_{\mathsf{max}} \cdot H.$$

The percentage with cap argument system above satisfies all the correctness and security properties that are specified in Section 2. We formally state these properties in the following theorems.

Theorem 3.1 (Completeness)**.** *The percentage with cap argument satisfies completeness 2.2.*

Theorem 3.2 (Soundness)**.** *Suppose that $\mathbb{G}$ is a prime order group for which the discrete log relation assumption (Definition 1.1) holds. Then the percentage with cap argument satisfies witness-extended emulation soundness 2.3.*

Theorem 3.3 (Zero-Knowledge)**.** *The percentage with cap argument satisfies perfect honest verifier zero-knowledge 2.4.*

The security proofs for the theorems above is straightforward from the completeness, soundness, and zero-knowledge properties of the commitment-commitment equality and the zero-balance argument systems. We provide an overview of proof in Section 4.

4 Proof of Security

The percentage with cap argument system is an OR-composition of the commitment-commitment equality argument and the zero-balance argument system. Therefore, the completeness, soundness, and zero-knowledge properties of the percentage with cap argument system follow straightforwardly from the security properties of the equality and zero-balance argument systems. Therefore, in this section, we provide the high-level overview of the proofs of the completeness, soundness, and zero-knowledge properties and defer the full details of proof to the documents of these individual argument systems.

Completeness. The completeness of the percentage-with-cap argument follows directly from the completeness of the two underlying argument systems it combines: the commitment-commitment equality proof and the zero-balance proof. The protocol ensures that a prover with a valid witness for one of the two conditions will always follow the honest protocol for that specific condition. The proof for the alternative, unsatisfied condition is perfectly simulated. Because the verifier's checks will pass for both the honestly generated proof and the simulated one, the entire OR-composition is accepted by the verifier, thus satisfying the completeness property.

Soundness. The soundness of the argument relies on the soundness of the underlying commitment-commitment equality and zero-balance arguments, which is secured by the discrete log relation assumption in the group. The soundness property requires that an efficient emulator $\mathcal{E}$ can extract a valid witness from any accepting transcript created by a cheating prover. The extractor for the OR-composed protocol leverages rewinding to obtain two accepting transcripts for the cheating prover with common commitments but different overall challenges c and c'. The verifier's logic computes the challenges for the individual proofs based on this overall challenge $c_{\mathsf{eq}} \leftarrow c - c_{\mathsf{max}}$. If the aggregated challenges c and c' differ, then the challenges for at least one of the individual protocols must also differ. The extractor can then focus on the specific protocol instances with differing challenges. By providing these two valid transcripts to the knowledge extractor for that

individual sigma protocol (either zero-balance or equality), it can correctly extract the corresponding witness.

Zero-Knowledge. The protocol achieves perfect honest-verifier zero-knowledge because a simulated transcript is indistinguishable from a real one. A simulator can produce a valid-looking transcript without knowing the actual witness by following the same logic as the prover's simulation strategy in the OR-composition.

- **Fee is less than the max fee**: The prover (and simulator) honestly generates the equality proof and simulates the zero-balance proof by choosing random values for the challenge and blinded value, then solving the verification equation.
- **Fee is equal to the max fee**: The prover (and simulator) honestly generates the zero-balance proof and simulates the equality proof by choosing random blinded values and solving for the commitment values that satisfy the verification equation.

In both scenarios, the final transcript is statistically identical to one generated from an honest interaction, as the verifier only sees a set of values that satisfy the required equations. This ensures that the verifier learns nothing about which of the two statements is actually true.

References

[1] Boneh, D., Drijvers, M., and Neven, G. Compact multi-signatures for smaller blockchains. In *International Conference on the Theory and Application of Cryptology and Information Security* (2018), Springer, pp. 435–464.

[2] Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., and Maxwell, G. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy (SP)* (2018), IEEE, pp. 315–334.

[3] Groth, J., and Ishai, Y. Sub-linear zero-knowledge argument for correctness of a shuffle. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (2008), Springer, pp. 379–396.

[4] Lindell, Y. Parallel coin-tossing and constant-round secure two-party computation. *Journal of Cryptology 16*, 3 (2003).